Multi-Objective Optimization for Cost and Latency in Computing Force Network

Shizhan Lan[1,2], Siyuan Song[2], Zhenyu Wang[2] and Yuxuan Long[2,*]
[1]China Mobile Guangxi Branch Co., Ltd, Nanning, China
[2]South China University of Technology, Guangzhou, China

Keywords: Computing Force Network, Workflow Scheduling, Price-Sensitive.

Abstract: Computing Force Network (CFN) is a new infrastructure based on cloud, edge and end three-layer network architecture. In CFN, tasks exist in the form of micro-services, so how to reduce the cost of user micro-services and ensure the quality of service is a challenging problem. In order to solve the above problems, firstly, we established the hierarchical model, resource limitation model, price model and time delay model of micro-service workflow. Secondly, we modeled the micro-service scheduling problem under the computing network into a multi-objective optimization problem with resource limitation as constraint and micro-service cost and overall time delay as targets. Thirdly, we propose a multi-objective optimization algorithm based on NSGA-II to solve the above problems. The experimental results show that the multi-objective optimization model established in this paper is effective, and the multi-objective optimization algorithm proposed in this paper is superior to the existing algorithms, which can effectively reduce the micro-service delay and cost.

1 INTRODUCTION

In recent years, with the development of technologies such as 5G and the Internet of Things, the amount of data in the global network has grown exponentially, and the constantly surging amount of data has brought huge challenges to cloud computing data centers. Edge computing is an emerging distributed computing paradigm. By extending the computing capacity of cloud data centers to the edge of networks, part of the data in the network can be processed by the edge, relieving the pressure on cloud data centers to some extent (J. Zhang, 2018). For example, from large cloud data centers to scattered Mobile Edge Computing (MEC) servers to mobile smart devices, the traditional network architecture has gradually evolved into cloud, edge, and end three-layer network architecture. In order to make better use of heterogeneous resources distributed in different places and realize accurate matching between user resource demand and heterogeneous resource supply, Computing Force Network is proposed.

In CFN, tasks submitted by users are usually divided into multiple micro-services and assigned to different compute nodes for processing. The compute nodes communicate and exchange data through network connections to achieve collaborative calculation and result summary (Islam A, 2021). In order to ensure the QoS of user tasks, the following aspects need to be considered.

Firstly, Cost is the key constraint. Due to the heterogeneity of the underlying hardware of the server and the interference of various factors, the rental price and cost of different servers vary greatly. However, users always hope that the rental cost will not exceed the cost budget when renting servers to deploy micro-services. Especially for high-performance computing micro-services, AI accelerator is expensive, and reasonable scheduling of micro-services can save costs for enterprises. Therefore, how to schedule micro service under strict price cost budget is the primary consideration (Tang X, 2022).

Secondly, Resource requirements are heterogeneous. Micro-services in computing networks have heterogeneous resource requirements. In addition to general resource requirements such as CPU and memory, many micro-services also require more dimensions of resource allocation according to specific business requirements. Therefore, the heterogeneity of micro-service resource requirements increases the complexity of micro-service scheduling.

Thirdly, the network becomes the bottleneck of application QoS. The scheduling problem of micro-services can be regarded as the scheduling of

Lan, S., Song, S., Wang, Z. and Long, Y.
Multi-Objective Optimization for Cost and Latency in Computing Force Network.
DOI: 10.5220/0012281700003807

In *Proceedings of the 2nd International Seminar on Artificial Intelligence, Networking and Information Technology (ANIT 2023)*, pages 291-300
ISBN: 978-989-758-677-4

workflow with dependency relationship. There is a sequence of execution among micro-services, and there is a relationship of data transmission between different micro-services. In the computing power network, computing resources are connected through the network. Optimizing the communication delay of micro-service is of great significance to reduce network congestion and ensure the application of QoS.

Our contributions are multifold and can be summarized as follows:

Firstly, considering the cost of micro-service scheduling, we set up a cost model from the perspective of user micro-service scheduling cost.

Secondly, we take into account the resource constraint of micro-service scheduling. Nodes that do not meet the resource constraint are not allowed to serve as scheduling nodes, so as to ensure the service quality of micro-service, which is different from previous workflow scheduling studies.

Then, we set up a multi-objective optimization model with resource constraints and cost and time delay as optimization objectives.

Finally, we propose a target capture optimization model based on NSGA-II to solve the above problems.

2 RELATED WORK

Many scholars have carried out in-depth research on the optimization of pricing cost and delay of micro-service scheduling.

According to the number of scheduling objectives, the current micro-service scheduling can be divided into single-objective optimization micro-service scheduling and multi-objective optimization micro-service scheduling. In the single-objective optimization micro-service scheduling, only one index is optimized, so the scheduling result is too limited. In the micro-service scheduling with multi-objective optimization, considering multiple constraints and optimization objectives, the scheduling results are more applicable. According to the types of micro-service scheduling, micro-service scheduling can be divided into mutually independent micro-service scheduling and workflow scheduling. The mutually independent micro-service scheduling does not consider the dependency between micro-services, while workflow scheduling considers the execution sequence of micro-services, and its scheduling implementation is more complex. Micro-service scheduling algorithm can be divided into heuristic scheduling algorithm and meta-heuristic scheduling algorithm.

For the delay problem of microservice scheduling, H. Topcuoglu proposed a Heterogeneous earliest-finisher (HEFT) algorithm and a Critical-Path-on-a-Processor, heterogeneous earlier-finisher (HEFT) algorithm (Topcuoglu H, 2002). In the CPOP algorithm, HEFT selects the task with the highest ascending rank value in each step and assigns the selected task to the processor, which minimizes its earliest completion time using the insertion-based method. In the CPOP algorithm, the priority of each task is calculated by comprehensively considering the ascending and descending sort. Since the above two algorithms were proposed, many scholars have proposed many improved algorithms based on the ideas of the above two algorithms according to different problem scenarios. Xiumin Zhou et al. proposed a heterogeneous earliest completion time (FDHEFT) algorithm based on fuzzy dominance sorting, which closely combines the fuzzy dominance sorting mechanism with the list scheduling heuristic HEFT, while optimizing the scheduling cost and delay (Zhou X, 2019). Faragardi et al. proposed a new resource supply mechanism and workflow scheduling algorithm GRP-HEFT, which is used to minimize the maximum completion time of a given workflow, so as to meet the budget constraints of the pay-as-the-volume cost model in modern IaaS cloud (Faragardi H R, 2020). In view of workflow scheduling problems, the above algorithms optimize the delay of workflow scheduling under the condition of satisfying workflow cost constraints. However, the above algorithms schedule with virtual machine as granularity, resulting in a large amount of resource waste. Moreover, the above algorithms do not consider the critical path of tasks as a whole, so it is easy to fall into local optimal. In order to implement global scheduling of micro-service, some scholars propose to use heuristic algorithm to solve micro-service scheduling problem. Lin et al. proposed an ant colony algorithm for solving scheduling problems, which not only considered the calculation of physical nodes and the utilization rate of storage resources, but also the number of micro-service requests and failure rate of physical nodes. Experimental results showed that the algorithm achieved better results in optimizing cluster business reliability, cluster load balancing and network transmission overhead(Lin M, 2019). Aiming at minimizing the cost of micro-service scheduling, Hussain et al proposed a hybrid cuckoo search and genetic algorithm HFSGA algorithm to realize micro-service scheduling (Hussain S M, 2022). But their approach is also

virtual machine granularity, resulting in a waste of resources. Liang et al. proposed an heuristic micro-service scheduling algorithm based on container to solve the scheduling problem of application workflow based on micro-service with minimum end-to-end delay under user-specified budget constraints (Bao L, 2019). This algorithm optimizes the delay of micro-service scheduling. His method takes container as unit for scheduling, which improves resource utilization in the scheduling process. However, it ignores the resource limitation of nodes. Once the resource allocation of containers exceeds the resource capacity of nodes, QoS applied by users will be seriously affected、. To sum up, the existing methods have some problems, such as too large granularity of micro-service scheduling, ignoring resource constraints and price and cost factors, which cannot meet the micro-service scheduling requirements under the computing power network.

3 SOLUTIONS

In the following, we first model the micro-service scheduling problem under the computing network, and then propose a heuristic algorithm based on NSGA-II to solve the above problem model.

3.1 Subsection System Model

This scenario consists of multiple physical server servers in geographically remote locations. The nodes are connected over the core network. Each node has heterogeneous resources, such as CPU, memory, bandwidth, and GPU. User tasks are broken down into multiple micro-services with dependencies. Micro-services exist in the form of containers. When users use node resources, they are rented as VMS. Users need to pay the VM rental fee. The fee is determined by the VM rental price per unit time and the VM rental duration. The execution of micro-services and data transmission between micro-services will result in a certain delay, and the overall delay of micro-services will affect the QoS of applications. When the micro-service is scheduled to a node, the amount of micro-service resources requested cannot exceed the remaining resources of the node. Before dispatching the service, the user will inform the cloud manufacturer of the price expectation and hope to obtain the highest QoS within the price expectation. The system model is shown in figure 1.

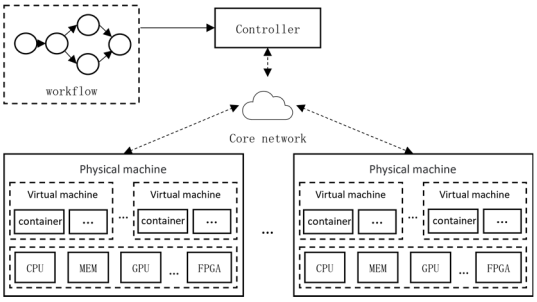


Figure 1. System model.

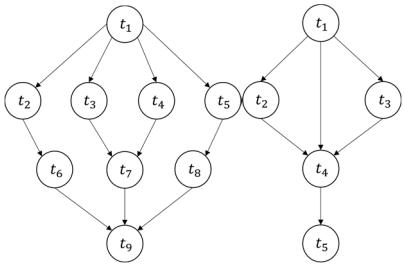


Figure 2. Workflow model.

3.2 Microservice Workflow Model

The workflow between microservices can be represented by a directed acyclic graph. Figure 2 describes the workflow structure of two microservices with dependencies.

For any microservice workflow, it can be expressed by $G(T,E)$, where $T=\{t_1,t_2,\ldots,t_m\}$ represents a set of microservices that are dependent on each other, $E=\{e_{i,j}|t_i,t_j\in T\}$ represents a direct dependency between microservices, If the microservice$t_j$depends on the microservice t_i, $e_{i,j}$ is 1, otherwise $e_{i,j}$ is 0. Set $pre(t_i)$ represent all precursor nodes of microservice t_i, $front(t_i)$ represents the direct precursor of microservice t_i, and use $after(t_i)$ to represent the direct successor of microservice t_i.

In this paper, the hierarchical workflow model mentioned in literature(Rizvi N, 2020) is used to process the above workflows. The specific process is to divide the DAG graph into multiple microservice chains. The entry microservice of each microservice chain has no precursor node, and the exit microservice has no successor node. According to the split microservice chain, the tasks are divided into different levels, and each level contains a set of independent microservices. For example, workflow 1 in Figure 4 can be divided into four microservice chains: t_1-t_2-t_6-t_9、t_1-t_3-t_7-t_9、t_1-t_4-t_7-t_9、t_1-t_5-t_8-t_9. In each microservice chain, microservice levels are divided. For example, in the t_1-t_2-t_6-t_9 chain, if

t_1 is the entry node, t_1 is the first layer, and accordingly t_2、t_6、t_9 are the second, third, and fourth layers respectively. Finally, for any microservice t_1, if t_1 is in multiple microservice chains, the levels of t_1 in each microservice chain are$rank_1$、$rank_1$、…、$rank_n$, the final t_i level is the highest of the preceding levels. For example, in workflow 2 in Figure 4, t_4 is in the microservice chain t_1-t_2-t_4-t_5、t_1-t_4-t_5 and t_1-t_3-t_4-t_5. The corresponding levels of t_4 in each chain are 2, 1, and 2 respectively, so the final level of t_4 is 2.

3.3 Resource Constraint Mode

Each VM has a certain amount of heterogeneous resources. When microservices are scheduled to VMS, they must meet the resource restrictions of VM nodes. Use $V = \{v_1, v_2, \dots, v_n\}$ to represent the set of all virtual machine nodes. For any virtual machine node v_j, there is a certain amount of heterogeneous resources. This paper considers four heterogeneous resources, namely CPU, memory, bandwidth, and GPU. R_{cpu}^j、R_{mem}^j、R_{net}^j、R_{gpu}^j are used to represent the remaining amount of four heterogeneous resources on the node at time t, respectively. The application amount of heterogeneous resources applied by microservice t_i is expressed by r_{cpu}^i、r_{mem}^i、r_{net}^j、r_{gpu}^i respectively. The 0-1 variable $z_{i,k}$ indicates whether the microservice t_i is scheduled to node v_k. When $z_{i,k}$ is 1, it indicates that the microservice t_1 is scheduled to v_k. When $z_{i,k}$is 0, it indicates that the microservice t_i is not scheduled to v_k. The resource cannot be preempted. The requested resource is released after the microservice is executed. The following constraints must be met during microservice scheduling

$$\sum_{t_i \in T, k=1}^{n} z_{i,k} = 1 \tag{1}$$

$$\sum_{\forall t_i \in T} r_{cpu}^i z_{i,k} \le R_{cpu}^k \tag{2}$$

$$\sum_{\forall t_i \in T} r_{mem}^i z_{i,k} \le R_{mem}^k \tag{3}$$

$$\sum_{\forall t_i \in T} r_{net}^i z_{i,k} \le R_{net}^k \tag{4}$$

$$\sum_{\forall t_i \in T} r_{gpu}^i z_{i,k} \le R_{gpu}^k \tag{5}$$

Formula (1) indicates that all microservices must be scheduled and can only be scheduled to one node at a time. Formulas (2) to (5) indicate that when microservices are scheduled to any node, the number of heterogeneous resources applied for microservices must be less than or equal to the remaining resources on the node.

3.4 Price-Cost Model

When a user rents a VM, the cloud vendor charges the user a fee based on whether the user rents the VM and the VM usage time, regardless of how many microservice containers the user schedules on the VM. The total price that the user needs to pay is represented by Cost, and the calculation formula of Cost is shown in formula (6).

$$Cost = \sum_{k=1}^{n} p_k \times t_{v_k} \tag{6}$$

t_v_k indicates the total duration of VM v_k rental, and p_k indicates the unit price of VM v_k rental. The user will submit a price budget before microservice scheduling, so the overall price cost should be lower than the budget after the final microservice execution is completed, otherwise the scheduling will fail. This paper deals with the budget by first optimizing the microservice delay and cost at the same time, finally getting the Pareto frontier, and then calculating the scheduling scheme with the lowest delay within the budget according to the price budget. Therefore, the final cost will be as close to the budget as possible, so as to obtain the best QoS within the budget. The price of a VM is related to the computing power of the VM per unit computing resource. Generally, the higher the price of a VM, the greater the computing power of the VM per unit computing resource, and the shorter the execution time of microservices.

3.5 Delay Model

In this paper, the time delay from the start of the first microservice to the end of the last microservice will be referred to as makespan, makespan is calculated as

$$makespan = \max_{\forall t_i \in T} FT(t_i) \tag{7}$$

Where $FT(t_i)$ represents the total time taken from the start of scheduling the first microservice to the completion of the microservice t_i, and formulas (7) calculate the total time taken for all microservices to

be completed. For any microservice t_i, $FT(t_i)$ consists of two parts: $WT(t_i)$, the waiting time required for task execution, and its own execution time $exc_{i,k}$, where $WT(t_i)$ and $FT(t_i)$ are calculated respectively.

$$WT(t_i) = \max_{\forall t_j \in pre(t_i)} \left(FT(t_j) + trans(t_j, t_i) \right) \quad (8)$$

$$FT(t_i) = WT(t_i) + exc_{i,k} \quad (9)$$

In the above formula, assuming that the microservice t_i is at layer n, $WT(t_i)$ represents the total time taken for the first layer N-1 microservices to complete. $trans(t_j, t_i)$ indicates the data transmission delay of the precursor node t_j of microservice t_i. If the containers of microservice t_i and microservice t_j are scheduled to the same VM, the delay is ignored. Otherwise, the delay is the ratio of the size of the data transfer between the two microservices to the average bandwidth allocated by the container in which the two microservices reside. Assuming that the amount of data transfer between microservice t_j and microservice t_i is $length_{j,i}$, the formula for calculating $trans(t_j, t_i)$ is:

$$trans(t_j, t_i) = \begin{cases} 0 \\ \dfrac{length_{j,i}}{(r_{net}^i + r_{net}^j)/2} \end{cases} \quad (10)$$

$exc_{i,k}$ indicates the execution delay required to schedule microservice t_i to node v_k. The delay is negatively correlated with the computing power per unit computing resource of the VM. This paper assumes that the delay is known.

3.6 Overall Model Design

Generally, the price of a virtual machine is related to the computing power per unit of computing resource of a virtual machine. The higher the price of a virtual machine, the greater the computing power per unit of computing resource of a virtual machine, and the shorter the execution time of a microservice. However, the size of virtual machine computing power and the price of virtual machine is not a constant proportion, under normal circumstances, the price of virtual machine is far more than doubled when the virtual machine computing power is doubled. Therefore, excessive pursuit of delay reduction will make the final price exceed the user's cost budget. Similarly, if only lower cost is required, the delay of the entire application will increase, affecting the QoS of the application. Therefore, the optimization direction of price cost and delay is not consistent, so that both objectives can be optimized, so that a relatively optimal scheduling scheme can be obtained under each price budget. Combined with the above problem description and the general model formula of multi-objective optimization introduced in Section 4.2, the above problem is modeled into a multi-objective optimization model in this paper, as shown below.

The objective function is:

$$Cost = Min \sum_{k=1}^{n} p_k \times t_{v_k} \quad (11)$$

$$makespan = Min \max_{\forall t_i \in T} FT(t_i)$$

The relevant constraints are:

$$\sum_{\forall t_i \in T, k=1}^{n} z_{i,k} = 1$$

$$\sum_{\forall t_i \in T} r_{cpu}^i z_{i,k} \leq R_{cpu}^k$$

$$\sum_{\forall t_i \in T} r_{mem}^i z_{i,k} \leq R_{mem}^k$$

$$\sum_{\forall t_i \in T} r_{net}^i z_{i,k} \leq R_{net}^k$$

$$\sum_{\forall t_i \in T} r_{gpu}^i z_{i,k} \leq R_{gpu}^k$$

In this paper, the process of solving the final scheduling scheme is divided into two steps: the first step is to obtain a set of uniformly distributed feasible solutions by solving the above multi-objective optimization model, that is, Pareto optimal front; The second step is to solve the optimal scheduling scheme according to the price budget set by the user.

3.7 Muti-Objective Optimization Algorithm Based on NSGA-II

Figure 3 shows the flow chart of the algorithm. The input of the algorithm is microservice set, virtual machine node set, and user price expectation, and the output of the algorithm is Pareto optimal frontier.

In NSGA-II algorithm, the common encoding methods include binary encoding, symbol encoding and real encoding. The traditional binary coding and decoding process is more troublesome, but the real coding reduces the complexity of calculation and improves the efficiency of operation. The goal of this paper is to schedule m microservice containers to be scheduled on n virtual machine nodes. Based on the characteristics of the problems studied in this paper, the real coding mode is selected, as shown in Figure 4. The numbers 1-9 represent the microservice to be scheduled, Node1 to Node5 represent the number of the VM node that can be scheduled, and the number corresponding to the server node number indicates

that the microservice is scheduled to the corresponding VM node. For example, ((1,4),(6,7),(2,3,9),(5,8)) indicates a possible initial solution. Resource constraints must be satisfied when generating the initial feasible solution. In order to generate the initial population, this paper randomly generates x initial solutions and the initial population $P = \{S_1, S_2, \ldots, S_x\}$.

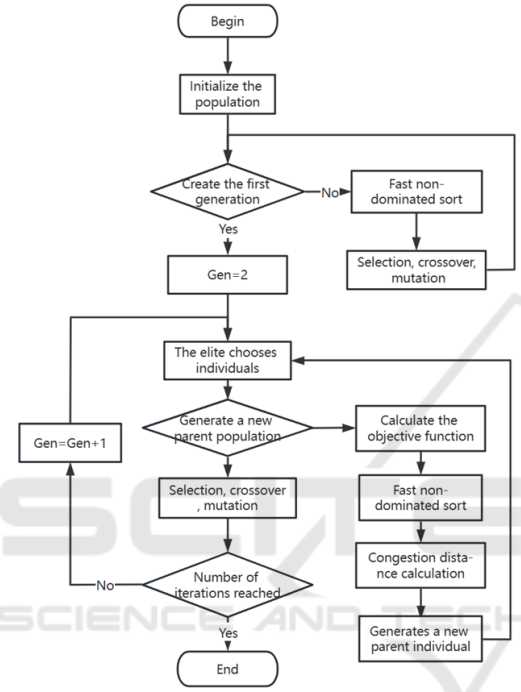


Figure 3. NSGA-II Algorithm Flowchart.

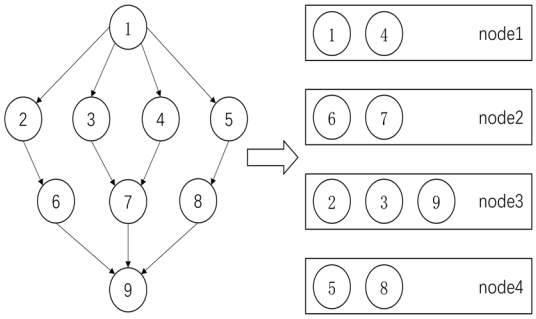


Figure 4. Coding Scheme.

The initial population was sorted according to the fitness function. The input of fast non-dominated sequencing was the original population P and the output was the stratified population P_r.

For the parent population $P = \{S_1, S_2, \ldots, S_x\}$. For any individual S_i, calculate the values of objective function 1 and objective function 2 of S_i. If for any other individual S_j in the population, S_j does not have a pareto dominance over S_i then divide S_i into the first non-dominated layer and traverse the population successively to find all the individuals meeting the above conditions. Divide all of the above individuals into the current tier and delete all of the above individuals into the current tier from the original population.

The number of non-dominant layers of the population is increased by one each time, and the above steps are repeated until there are no individuals in the original population, and finally the stratified population $P_r = \{\{S_1, S_x\}, \ldots, \{S_2\}\}$ .The number of non-dominated levels of an individual represents the quality of the solution, and the smaller the number of non-dominated levels, the better the performance of the individual and the closer to the optimization goal.

The crowding degree i_d of each individual in each layer of the stratified population P_r was calculated in turn. The degree of crowding represents the density of individuals around an individual in the population, and the value is equal to the circumference of the rectangle with the vertex near the point. Let the crowding degree of individuals O_d and I_d at the boundary position be ∞ , and the formula for calculating the crowding degree of individuals at the other positions be

$$i_d = \sum_j^m \left(\frac{\left| f_j^{i+1} - f_j^{i-1} \right|}{\left| f_j^{max} - f_j^{min} \right|} \right) \quad (12)$$

m is the number of fitness evaluation functions, f_j^{i+1} and f_j^{i-1} represent the function value of the JTH objective of the i+1 individual and the I-1 individual, respectively, f_j^{max} and f_j^{min} represent the maximum and minimum objective function values of all individuals in the current level for the objective j, respectively.

The elite selection strategy is based on non-dominant ordering and crowding distance to obtain progeny populations. Suppose that for the stratified population $P_r = \{\{S_1, S_x\}, \ldots, \{S_2\}\}$, each layer is sorted in ascending order by crowding distance, and the steps selected by the elite are: All the individuals from the first layer in P_r were added to the new population P, and then all the individuals from the second layer were added to the new population P, and so on, until the individuals from a certain layer could not all be added to the new population P, and the individuals from that layer were added to P in the order of the crowding degree distance, until the

number of individuals in the new population P reached x, as shown in Figure 5.

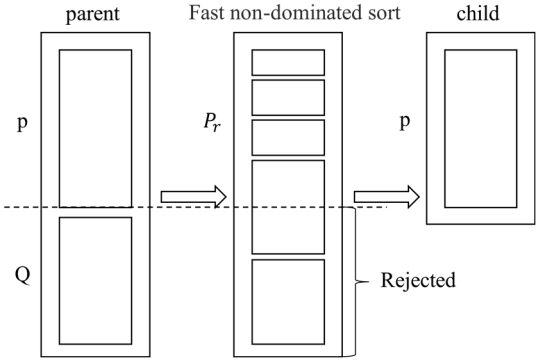


Figure 5. Fast non-dominated sorting algorithm

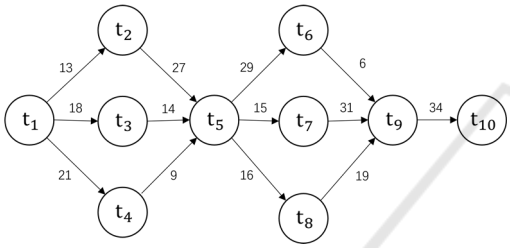


Figure 6. Workflow

4 EXPERIMENT

Table 1. Specifications and Prices of the Six VMS.

Instance	CPU	Memory	GPU	Bandwidth	cost
v_1	4	16	4	100	30.39
v_2	4	16	8	100	60.94
v_3	8	32	16	100	130.88
v_4	6	32	24	100	150.09
v_5	8	40	32	100	180.04
v_6	16	40	32	100	190.10

Table 2. NSGA-II Parameter Setting.

Parameter	number
Size	50
Number of iterations	50
Cross probability	1
Mutation probability	0.1

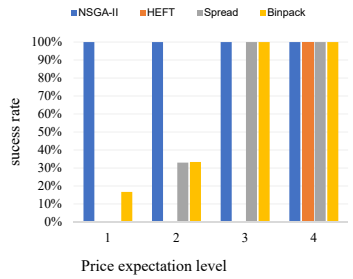
In this section, the NSGA-II-based microservice scheduling algorithm is tested using the Cloudsim simulation platform, which has the modeling and simulation functions of physical machines and containers. This paper first analyzes several common workflow structures in Alibaba Cluster Trace Program, and constructs DAG graphs with 5, 10, 15 and 20 microservices respectively by referring to common workflow structures. Figure 6 shows the DAG diagram when the number of microservices is 10, and the weights on the edges of the diagram represent the size of the data transfer volume of the microservices with dependencies.

The specifications and prices of the VMS used in the experiment refer to the cloud vendor's charging by volume rules. Table 1 lists the specifications and prices of the six VMS used in this paper.

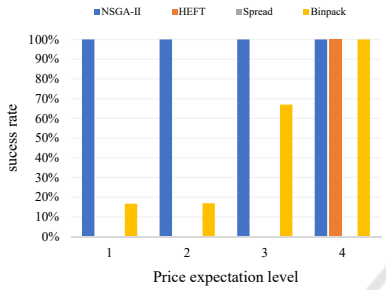
The parameter Settings of an algorithm largely determine the performance of the algorithm. Table 2 lists the parameters of the algorithm in this paper.

Price and microservice delay are used as evaluation indexes for microservice scheduling. The calculation formulas for the above two are formula (6) and formula (7) respectively. To verify the performance of the scheduling algorithms in this paper, the Spread, Binpack, and HEFT algorithms are selected as benchmarks. Spread and Binpack algorithms are common methods in container scheduling. Spread tends to distribute containers to each node to balance cluster load, while Binpack tends to dispatch containers to one node to improve resource utilization. HEFT algorithm is a classic algorithm in workflow scheduling. Its idea is to always schedule tasks to the node with the minimum completion time. However, HEFT algorithm schedules tasks based on virtual machines and does not consider resource constraints during scheduling. The HEFT is changed to a HEFT algorithm that schedules by container and considers resource constraints.

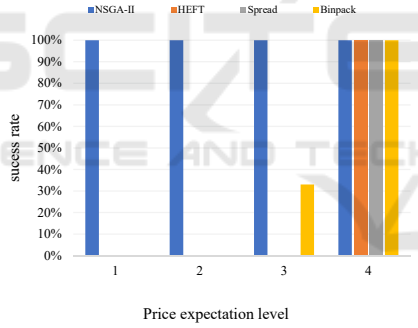
Figure 7 shows the scheduling success rates of the four algorithms at different price expectation levels. Subgraphs (a), (b), (c) and (d) respectively show the scheduling success rates of four algorithms with DAG sizes of 5, 10, 15 and 20. The higher the price expectation level, the more adequate the price budget given by the user. Because the Spread scheme tends to schedule microservices to different nodes, a large number of virtual machines are rented, and the data transmission delay between microservices becomes longer, which ultimately makes scheduling impossible under the condition of meeting the price constraint. The Binpack scheme tends to schedule microservices to a node, so the number of leased virtual machines is small and the communication delay between microservices is reduced, which can meet the price constraint to a certain extent, but it



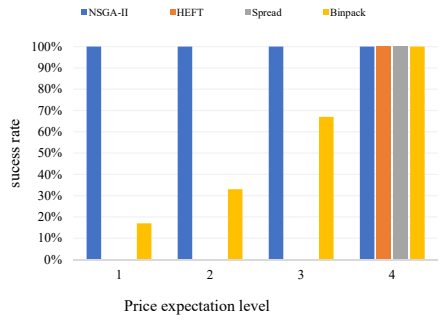
(a) 5 microservices



(b) 10 microservices



(c) 15 microservices

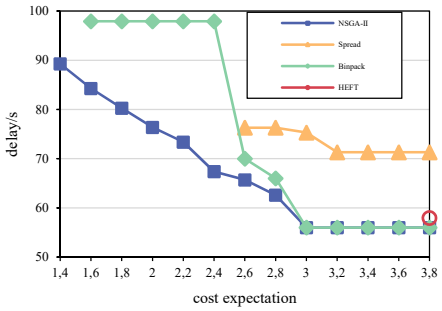


(d) 20 microservices

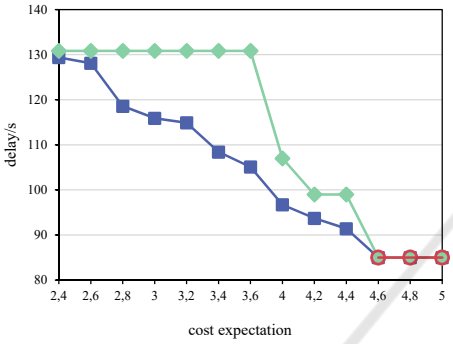
Figure 7. Success Rate.

cannot take into account the global scheduling, so it cannot meet the scheduling demand when the price constraint level is high. The HEFT algorithm always schedules microservices to the node with the shortest completion time, without considering the global scheduling and scheduling cost, so the scheduling result is difficult to meet the cost expectation set by users. The NSGA-II-based microservice scheduling algorithm proposed in this paper also optimizes the scheduling delay and cost of microservices, so the scheduling success rate of the algorithm proposed in this paper exceeds other algorithms.

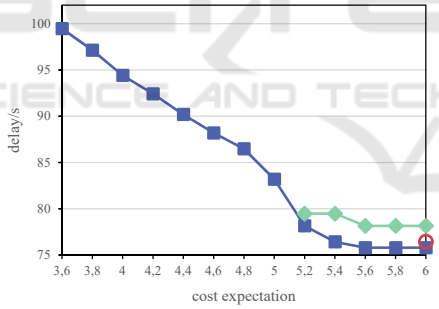
Figure 7 shows the scheduling delay of four microservice algorithms under four microservice scales and different price expectations. Subfigures (a), (b), (c), and (d) show the experimental results when DAG scales are 5, 10, 15, and 20 respectively. It can be seen from the figure that with the increase of price expectation, the delay of various algorithms shows a non-increasing trend. Among them, the algorithm proposed in this paper can obtain lower microservice delay compared with other algorithms under the same price expectation. When the price expectation increases to a certain value, the delay will no longer decrease, and higher QoS can no longer be obtained when the price expectation is increased. The above phenomenon is in line with normal logic, because microservice execution and data transmission will certainly cost a certain delay, and the computing power of virtual machine nodes and the transmission capacity of the network are limited, so the delay can not be reduced. From the experimental results, it can be seen that the algorithm proposed in this paper can find the global relative optimal scheduling scheme under the user-set price expectation. For example, under 5 microservices, when the user price expectation is 2, only the algorithm in this paper and the Binpack algorithm can give the scheduling scheme under the price expectation, and the other two algorithms fail to schedule. When the user's price expectation is 2.6, the algorithm in this paper obtains a lower delay than Binpack and Spread, so the user pays the same price, and the algorithm in this paper can obtain higher QoS.



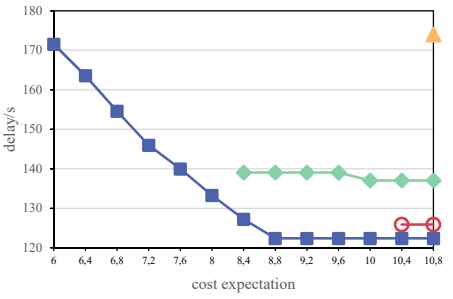
(a) 5 microservices



(b) 10 microservices



(c) 15 microservices



(d) 20 microservices

Figure 8. Scheduling delay in different DAGs.

In summary, the multi-objective optimization algorithm proposed in this paper can optimize both delay and price, and can select the relatively optimal scheduling scheme from the Pareto frontier solution according to the price expectation set by users, which improves the scheduling success rate and reduces the microservice delay. The algorithm proposed in this paper provides a solution for the price-sensitive microservice scheduling under the computing power network.

5 CONCLUSION

This paper first analyzes the microservice scheduling problem under the CFN, then we introduce the relevant theories and technologies of multi-objective optimization, and models the scheduling problem of microservice under the computing network into a multi-objective optimization problem. Finally, a multi-objective optimization algorithm based on NSGA-II is proposed to solve the above problem model. The experimental results show that the proposed algorithm can optimize both the price cost and the microservice delay, and finally give a relatively optimal scheduling scheme according to the price expectation set by the user.

The microservice scheduling model constructed in this paper does not take into account the oversold problem of resources, that is, the amount of resource applications of containers on a virtual machine node can be greater than the total amount of resources owned by the virtual machine. Further research can be carried out in the future.

REFERENCES

J. Zhang, B. Chen, Y. Zhao, X. Cheng and F. Hu, "Data Security and Privacy-Preserving in Edge Computing Paradigm: Survey and Open Issues," in *IEEE Access*, vol. 6, pp. 18209-18237, 2018, https://doi.org/10.1109/ACCESS.2018.2820162

Islam A, Debnath A, Ghose M, et al. A survey on task offloading in multi-access edge computing (J). *Journal of Systems Architecture*, 2021, 118: 102225. https://doi.org/10.1016/j.sysarc.2021.102225

Tang X, Cao W, Tang H. Cost-Efficient Workflow Scheduling Algorithm for Applications With Deadline Constraint on Heterogeneous Clouds (J). *IEEE Transactions on Parallel and Distributed Systems*, 2022, 33(9): 2079–2092. https://doi.org/10.1109/TPDS.2021.3134247

Topcuoglu H, Hariri S, Society I C. Performance-effective and low-complexity task scheduling for heterogeneous

computing (J). *IEEE Transactions on Parallel and Distributed Systems*, 2002, 13(3): 260–274. https://doi.org/10.1109/71.993206

Zhou X, Zhang G, Sun J. Minimizing cost and makespan for workflow scheduling in cloud using fuzzy dominance sort based HEFT (J). *Future Generation Computer Systems*, 2019, 93: 278–289. https://doi.org/10.1016/j.future.2018.10.046

Faragardi H R, Reza M, Sedghpour S. GRP-HEFT : A Budget-Constrained Resource Provisioning Scheme for Workflow Scheduling in IaaS Clouds (J). *IEEE Transactions on Parallel and Distributed Systems*, 2020, 31(6): 1239–1254. https://doi.org/10.1109/TPDS.2019.2961098

Lin M, Xi J, Bai W. Ant Colony Algorithm for Multi-Objective Optimization of Container-Based Microservice Scheduling in Cloud (J). *IEEE Access*, 2019, 7: 83088–83100. https://doi.org/10.1109/ACCESS.2019.2924414

Hussain S M, Begh G R. Hybrid heuristic algorithm for cost-efficient QoS aware task scheduling in fog–cloud environment (J). *Journal of Computational Science*, Elsevier B.V., 2022, 64(January): 101828. https://doi.org/10.1016/j.jocs.2022.101828

Bao L, Wu C, Bu X, et al. Performance Modeling and Workflow Scheduling of Microservice-Based Applications in Clouds(J). *IEEE Transactions on Parallel and Distributed Systems*, IEEE, 2019, 30(9): 2101–2116. https://doi.org/10.1109/TPDS.2019.2901467

Rizvi N, Ramesh D. HBDCWS: heuristic-based budget and deadline constrained workflow scheduling approach for heterogeneous clouds (J). *Soft Computing, Springer Berlin Heidelberg*, 2020, 24(24): 18971–18990. https://doi.org/10.1007/s00500-020-05127-9